Réalisation d'une application de gestion de production: Rapport de projet

Pierre Lison (étudiant INFO 22)
plison@student.fsa.ucl.ac.be

Université Catholique de Louvain
Faculté des Sciences Appliquées
Département d'ingénierie informatique

31 août 2004

Table des matières

1. Introduction et cahier de charges

1.1 Objectif du présent rapport

Ce rapport, rédigé dans le cadre du projet du cours LINF 2172 "Conception de bases de données", poursuit un double objectif :

1. Détailler la **méthodologie** utilisée dans la conception et l'implémentation de l'application ;
2. Préciser, à chaque étape, les **résultats** obtenus, et particulièrement le fonctionnement de l'application finale.

Le prototype de l'application est disponible à cette adresse :
`http ://www2.awake2life.com/vm008/tiki-db2.php`

Celle-ci a été "déployée" sur le site de mon unité scoute, qui disposait en l'occurence de l'architecture MySQL-Apache-PHP nécessaire au bon fonctionnement de l'application.

1.2 Cahier de charges

1.2.1 Exigences sur la base de données

Il nous est demandé de concevoir une base de données destinée à la gestion de la production et des stocks d'une société. Pour ce faire, une description "narrative" de celle-ci est fournie :

> "Une *unité de fabrication* est identifiée par un numéro et caractérisée par un nom, une localisation et une capacité globale de production. Elle peut fabriquer plusieurs produits. A tout moment, on connaît, pour une unité de fabrication et pour un produit que cette unité fabrique, la quantité en cours de fabrication et les capacités minimum et maximum de fabrication.
> Un *dépôt* est également identifié par un numéro et possède un nom, une localisation et une capacité globale de stockage. Un dépôt peut stocker plusieurs produits. A chaque instant, on connaît, pour un dépôt et pour un produit qui y est stocké, la quantité due.
> Un *produit* peut être fabriqué par plusieurs unités de fabrication et socké dans plusieurs dépôts. A chaque produit est associé un numéro qui l'identifie. Un produit possède également un nom, une unité de mesure et un prix. Une quantité unité de produit fini ou semi-fini est toujours fabriquée à partir de quantités déterminées d'autres produits semi-finis et/ou de matières premières qu'une unité de fabrication commande à ce dépot.
> Une *commande* d'une unité de fabrication est adressée à un seul dépôt, est identifiée par un numéro, possède une date et concerne un ou plusieurs produits dont elle spécifie la quantité commandée."

1.2.2 Exigences sur les transactions

Au point de vue des transactions, deux exigences sont spécifiées :

1. "Interfacer l'application de telle manière à ce qu'un utilisateur puisse insérer / supprimer / modifier des données"
2. "Programmer les requêtes suivantes :
   (a) Donner le nom des unités de fabrication ainsi que l'identifiant des lignes de commandes d'un produit
   (b) Donner le montant total (en EUR) associé aux commandes réalisées pour un produit donné durant un certain intervalle de temps (entre deux dates).
   (c) Donner l'identifiant et le nom des produits et/ou matières premières qui composent un produit donné"

1.3 Limitations

Etant donné la taille du projet (initialement conçu pour un groupe de 5 personnes), il ne nous a pas été paru utile d'implémenter l'ensemble des exigences sur les transactions. Nous avons préféré nous concentrer sur l'essentiel, sur ce qui constituait à nos yeux le coeur du projet : une **modélisation rigoureuse** de la base de données et des transactions correspondantes.

Plus précisément, voici les améliorations que nous pourrions apporter à l'application existante pour respecter complètement les exigences du projet :

1. Vérification systématique de certaines contraintes sémantiques (telle que la production actuelle comprise entre la production minimum et maximum, la quantité de produit en stock plus petite que la capacite totale de stockage,...). Comme MySQL ne traite pas directement ces assertions, il faut nous en charger nous-mêmes au niveau du "langage hôte", c'est-à-dire PHP.

2. Vérification des valeurs "vides" ou incorrectes lors d'ajout ou de modification de données.

3. Conception de formulaires précis et complets pour l'ajout, la modification et la suppression de données pour toutes les entités de la base de données (*Actuelleemnt, seul le formulaire d'ajout de produits est implémenté, à titre d'exemple*)

4. Amélioration générale de l'interface graphique (suppression de la décoration scoute, mise en page plus ergonomique,...)

5. Mise en place de liens à l'intérieur des tableaux permettant d'accéder à une information sur un tuple spécifique.

6. Ecriture de spécifications formelles ou semi-formelles pour l'ensemble du code.

7. Conception d'une batterie de tests complète pour vérifier les exigences.

8. Ecriture d'un mode d'emploi.

9. etc.

2. Modélisation ER de la base de données

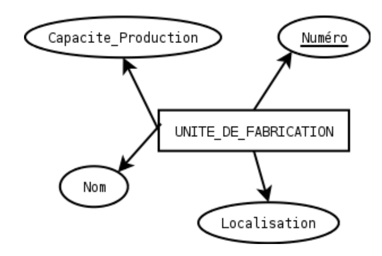
2.1 Modélisation des entités et attributs

La première étape dans la conception de notre modèle ER (Entity-Relationship) est la découverte des *entités* - "objets" du monde réel, dotés d'une existence indépendante - et de leurs *attributs* respectifs - les propriétés qui caractérisent ces dernières.

A partir de la description qui nous est fournie, nous pouvons d'ores et déjà épingler certains concepts qui nous faudra modéliser par des entités :

2.1.1 UNITE DE FABRICATION

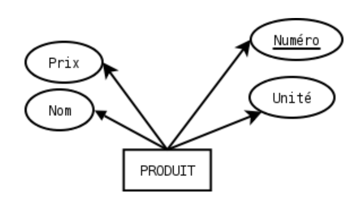
Voici le schéma de l'entité `UNITE DE FABRICATION` qui satisfait aux exigences du cahier des charges détaillé à la section précédente :



Chaque unité est donc composée :
- d'un *numéro* : un entier, **clé primaire** de l'entité (la relation de l'unité à son numéro est injective).
- d'un *nom* : une chaîne de caractères.
- d'une *capacité maximale de production* : un réel.
- d'une *localisation* : une chaîne de caractères.

2.1.2 PRODUIT

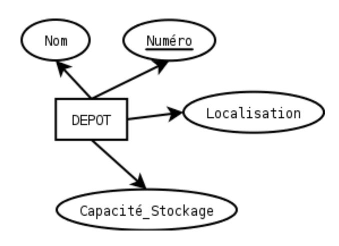
Voici le schéma de l'entité `PRODUIT` qui satisfait aux exigences du cahier des charges :



Chaque produit est donc composé :
- d'un *numéro* : un entier, **clé primaire** de l'entité (la relation du produit à son numéro est injective).
- d'un *nom* : une chaîne de caractères.
- d'un *prix* : un réel
- d'une *unité de mesure* : une chaîne de caractères.

2.1.3 DEPOT

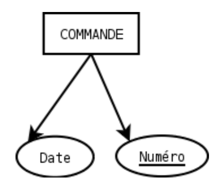
Voici le schéma de l'entité `DEPOT` qui satisfait aux exigences du cahier des charges :



Chaque dépot est donc composé :
- d'un *numéro* : un entier, **clé primaire** de l'entité (la relation du dépot à son numéro est injective).
- d'un *nom* : une chaîne de caractères.
- d'une *capacité maximale de stockage* : un réel.
- d'une *localisation* : une chaîne de caractères.

2.1.4 COMMANDE

Voici le schéma de l'entité `COMMANDE` qui satisfait aux exigences du cahier des charges :



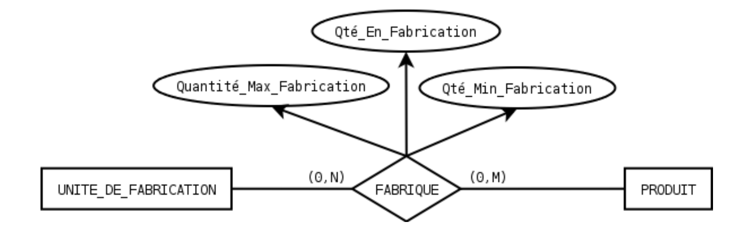
Chaque commande est donc composée :
- d'un *numéro* : un entier, **clé primaire** de l'entité (la relation de la commande à son numéro est injective)
- d'un *nom* : une chaîne de caractères.
- d'une *date* : de type `date` !

2.2 Modélisation des relations

Intéressons-nous à présent aux **relations** entre les différentes entités. Nous utiliserons la notation *(min, max)* pour les contraintes structurelles.

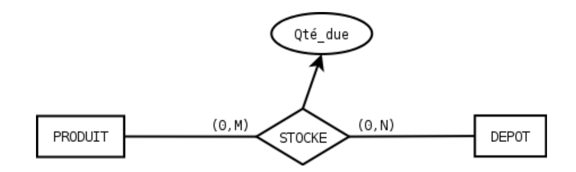
2.2.1 FABRIQUE

`FABRIQUE` relie `UNITE DE FABRICATION` à `PRODUIT`. Une unité peut fabriquer 0...N produits, et un produit peut être fabriqué à 0...M unités différentes. Trois attributs sont propres à la relation : *Qté min. de fabrication*, *Qté max. de fabrication*, et *Qté de fabrication* (qui correspond à la quantité actuelle), tous trois des réels.



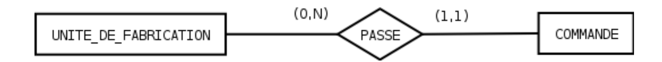
2.2.2 STOCKE

`STOCKE` relie `DEPOT` à `PRODUIT`. Un dépot peut stocker 0...N produits, et un produit peut être stocké à 0...M dépots différents. Un attribut de relation est défini : *Qté due*, un réel..



2.2.3 PASSE

`PASSE` relie `UNITE DE FABRICATION` à `COMMANDE`. Une unité peut passer 0...N commandes, mais une commande est toujours passée par une et une seule unité de fabrication (la participation est donc totale).



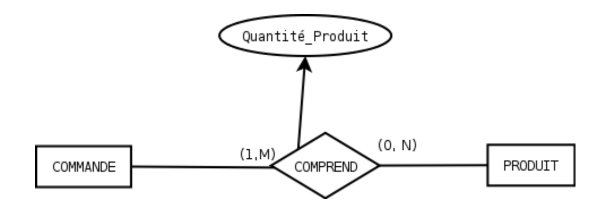
2.2.4 FOURNIT

`FOURNIT` relie `DEPOT` à `COMMANDE`. Un dépot peut fournir 0...N commandes, mais une commande est toujours fournie par un et un seul dépôt (la participation est donc totale).



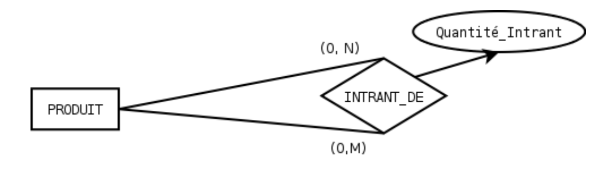
2.2.5 COMPREND

`COMPREND` relie `COMMANDE` à `PRODUIT`. Une commande peut contenir 1...N produits (la description indique qu'une commande doit contenir au moins un produit, la participation est donc totale), et un produit peut faire partie de 0...M commandes. Un attribut est définit : *Quantité produit*, un réel.



2.2.6 INTRANT DE

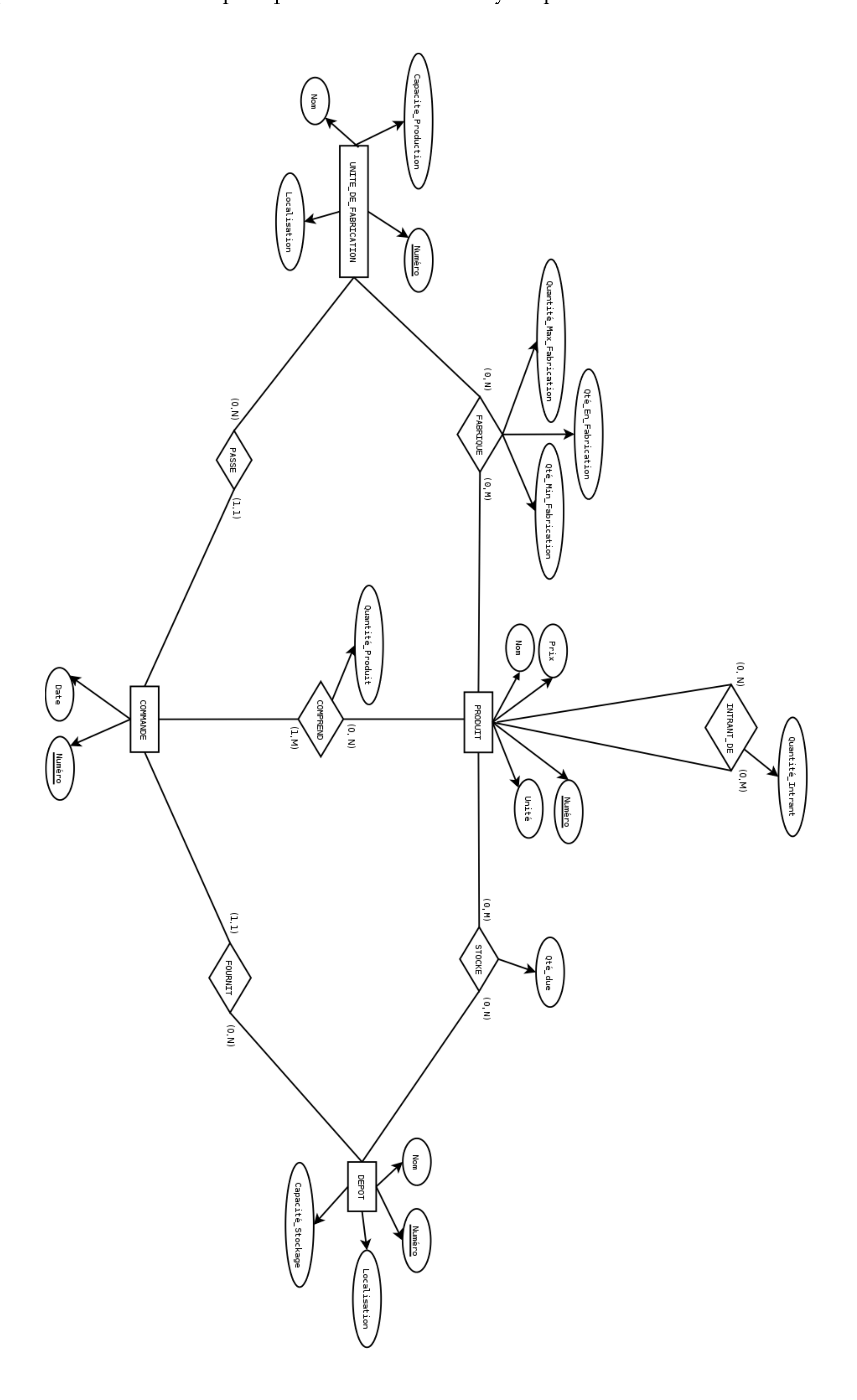
`INTRANT DE`[1] relie `PRODUIT` à `PRODUIT`. Un produit (matière première, produit semi-fini,...) peut engendrer 0...N autres produits, et un produit peut être engendré par 0...M produits. Un attribut est défini : *Qté intrant*, un réel qui correspond à la quantité d'intrant nécessaire pour la fabrication d'une quantité unitaire du produit sortant.



[1]Intrant est la traduction française de "input"

2.3 Vue d'ensemble

Voici à présent le schéma complet qui résulte de nos analyses précédentes :



2.4 Alternatives et raffinements possibles

Le modèle ER actuel peut être amélioré de différentes manières :

- On pourrait appliquer une *spécialisation* à l'entité `PRODUIT` en y distinguant les matières premières, les produits semis-finis et produits finis. Ce n'est pour l'instant pas nécessaire car la description fournie ne précise aucun attribut qui leur serait particulier. Néammoins, si la cahier des charges venait à changer dans le sens d'une telle distinction, cette spécialisation pourrait être utile.

- Pour modéliser le concept de *commande*, nous avons aussi pensé à une relation ternaire. Mais un problème s'est posé pour le lien entre `COMMANDE` et `PRODUIT` : il est en effet difficile d'y noter explicitement qu'une commande peut contenir plusieurs produits. Notre choix de représente une commande par une entité distincte est donc préférable.

3. Modélisation relationelle de la base de données

Une fois la modélisation ER terminée, nous passons à la traduction de celle-ci en *modèle logique*, qui, dans notre cas, est un modèle relationnel. Nous utiliserons ici la méthodologie présentée au chapitre 7 du livre d'Elmasri et Navathe[1].

3.1 Traduction des entités régulières

3.1.1 UNITE DE FABRICATION

```
CREATE TABLE UNITE_DE_FABRICATION
(FAB_NUM INT NOT NULL,
NOM VARCHAR(15),
LOCALISATION VARCHAR(30),
CAPA_PROD REAL,
PRIMARY KEY (FAB_NUM));
```

3.1.2 PRODUIT

```
CREATE TABLE PRODUIT
(PROD_NUM INT NOT NULL,
NOM VARCHAR(15),
UNITE VARCHAR(10),
PRIX REAL NOT NULL,
PRIMARY KEY (PROD_NUM));
```

3.1.3 DEPOT

```
CREATE TABLE DEPOT
(DEP_NUM INT NOT NULL,
NOM VARCHAR(15),
LOCALISATION VARCHAR(30),
CAPA_STOCK REAL,
PRIMARY KEY (DEP_NUM));
```

3.1.4 COMMANDE

Ici, puisque nous avons deux relations 1 :N (la relation `PASSE` entre `UNITE DE FABRICATION` et `COMMANDE`, et la relation `FOURNIT` entre `DEPOT` et `COMMANDE`), nous devons inclure dans la table deux **clés étrangères**, `FAB NUM` et `DEP NUM`, le numéro de l'unité de fabrication et le numéro du dépot (respectivement) qui correspondent à la commande.

```
CREATE TABLE COMMANDE
(COM_NUM INT NOT NULL,
COM_DATE DATE NOT NULL,
FAB_NUM INT NOT NULL,
DEP_NUM INT NOT NULL,
PRIMARY KEY (COM_NUM),
FOREIGN KEY (FAB_NUM) REFERENCES UNITE_DE_FABRICATION (FAB_NUM),
FOREIGN KEY (DEP_NUM) REFERENCES DEPOT (DEP_NUM));
```

[1] Elmasri and Navathe, *Fundamentals of Database Systems*, Addison-Wesley, 2004

3.2 Traduction des relations N :M

Pour chaque relation de type N :M, nous devons construire une nouvelle table incluant comme clés étrangères les clés primaires des entités participantes. Nous avons quatre relations de ce type :

3.2.1 PRODUIT FABRIQUE

Chaque tuple de cette table représente un produit particulier fabriqué dans une unité particulière. Elle inclut donc deux clés étrangères, et la clé primaire de cette table est la combinaison de ces deux clés. De plus, nous y incluons les trois attributs de la relation `FABRIQUE`.

```
CREATE TABLE PRODUIT_FABRIQUE
(FAB_NUM INT NOT NULL,
PROD_NUM INT NOT NULL,
MAX_FAB REAL,
MIN_FAB REAL,
FAB_ACTUELLE REAL NOT NULL,
PRIMARY KEY (FAB_NUM, PROD_NUM),
FOREIGN KEY (FAB_NUM) REFERENCES UNITE_DE_FABRICATION (FAB_NUM),
FOREIGN KEY (PROD_NUM) REFERENCES PRODUIT (PROD_NUM));
```

3.2.2 PRODUIT STOCKE

Chaque tuple de cette table représente un produit particulier stocké dans un dépot particulier. Elle inclut donc deux clés étrangères, et la clé primaire de cette table est la combinaison de ces deux clés. De plus, nous y incluons l'attribut `QUANTITE DUE` de la relation `STOCKE`.

```
CREATE TABLE PRODUIT_STOCKE
(DEP_NUM INT NOT NULL,
PROD_NUM INT NOT NULL,
QUANTITE_DUE REAL NOT NULL,
PRIMARY KEY (DEP_NUM, PROD_NUM),
FOREIGN KEY (DEP_NUM) REFERENCES DEPOT (DEP_NUM),
FOREIGN KEY (PROD_NUM) REFERENCES PRODUIT (PROD_NUM));
```

3.2.3 CONTENU COMMANDE

Chaque tuple de cette table représente un produit particulier compris dans une commande particulière. Elle inclut donc deux clés étrangères, et la clé primaire de cette table est la combinaison de ces deux clés. De plus, nous y incluons l'attribut `QUANTITE` de la relation `COMPREND`.

```
CREATE TABLE CONTENU_COMMANDE
(COM_NUM INT NOT NULL,
PROD_NUM INT NOT NULL,
QUANTITE REAL NOT NULL,
PRIMARY KEY (COM_NUM, PROD_NUM),
FOREIGN KEY (COM_NUM) REFERENCES COMMANDE (COM_NUM),
FOREIGN KEY (PROD_NUM) REFERENCES PRODUIT (PROD_NUM));
```

3.2.4 TRANSFORMATION

Chaque tuple de cette table représente un intrant particulier permettant de donner un produit sortant particulier. Elle inclut donc deux clés étrangères, et la clé primaire de cette table est la combinaison de ces deux clés. L'attribut de relation `QUANTITE` précise la quantité d'intrant nécessaire pour fabriquer une quantité unitaire de produit sortant.

```
CREATE TABLE TRANSFORMATION
(INTRANT INT NOT NULL,
SORTANT INT NOT NULL,
QUANTITE_INTRANT REAL NOT NULL,
PRIMARY KEY (INTRANT, SORTANT),
FOREIGN KEY (INTRANT) REFERENCES PRODUIT (PROD_NUM),
FOREIGN KEY (SORTANT) REFERENCES PRODUIT (PROD_NUM));
```

4. Implémentation de la base de données

4.1 Intégrité référentielle

MySQL ne vérifiant pas automatiquement les contraintes d'intégrité référentielle, nous avons dû procéder à la création d'`index` pour l'ensemble des clés étrangères, ce qui nous a permis, en spécifiant `InnoDB` comme *type* de la table, d'exiger le respect des contraintes d'intégrité référentielle.

4.2 Contraintes sémantiques

Certaines contraintes, comme le non-dépassement de la capacité maximale de production pour les unités de fabrication, ou de la capacité maximale de stockage pour les dépots, doivent également être vérifiées. MySQL ne permettant pas de spécifier directement celles-ci, la seule manière de procéder est de les implémenter au niveau du langage hôte : PHP. Cette implémentation étant particulièrement fastidieuse à mettre en place correctement, nous ne l'avons pas réalisée en pratique dans notre application.

Voici la liste des contraintes :

1. `Qté min. de fabrication` $\leq$ `Qté en fabrication` $\leq$ `Qté max. de fabrication`, pour un produit fabriqué donné
2. La somme des `Qtés en fabrication` doit être $\leq$ à la capacité max. de production, pour une unité de fabrication donnée.
3. La somme des `Qtés dues` doit être $\leq$ à la capacité maximale de stockage, pour un dépot donné.

5. Conception et implémentation des transactions

Le code source PHP de notre application est inséré en annexe. A titre d'exemple, nous en fournissons et expliquons ici un fragment :

```
if (isset ( $_REQUEST["afficher_usines"])){

     $query = "select * from ‘USINE‘";
     $listing_complet = $tikilib->query($query);

     $tableau_usine = "<table cellspacing = 1pt cellpadding=10pt border=1pt>
                      <tr>
                      <td><b>NUMERO</b></td>
                      <td><b>NOM</b></<td>
                      <td><b>LOCALISATION</b></td>
                      <td><b>CAPACITE TOTALE DE PRODUCTION</b></td>
                      </tr>";

     while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {

          $tableau_usine=$tableau_usine."<tr>
                         <td>".$ligne["FAB_NUM"]."</td>
                         <td>".$ligne["NOM"]."</td>
                         <td>".$ligne["LOCALISATION"]."</td>
                         <td>".$ligne["CAPA_PROD"]."</td></tr>";
     }

     $tableau_usine=$tableau_usine."</table>";
     $smarty->assign ('tableau', $tableau_usine);
}
```

Quelques commentaires sur celui-ci :

– La première et deuxième ligne à l'intérieur de la structure `if` expriment la requête SQL. `Tikilib` est une librairie nous permettant d'accéder facilement à la base de données.
– La troisième ligne construit le début du tableau HTML.
– La boucle `while` parcourt incrémentalement l'ensemble des tuples renvoyés par la requête SQL
– Pour chaque tuple, nous l'insérons dans notre chaîne de caractères `tableau usine`[1], décoré des balises HTML nécessaires.
– Une fois l'ensemble des tuples parcouru, nous fermons le tableau, et nous l'insérons à l'intérieur de la page (`Smarty` est un moteur de templates)

[1]Nous avons ici utilisé le terme "usine" en lieu et place d'"unité de fabrication", plus long à transcrire

A. Fichier `tiki-db2.php`

```
<?php
// $Header: /cvsroot/tikiwiki/tiki/tiki-remind_password.php,
//v 1.8.2.2 2004/03/28 09:59:53 damosoft Exp $

// Copyright (c) 2002-2004, Luis Argerich, Garland Foster, Eduardo Polidor, et. al.
// All Rights Reserved. See copyright.txt for details and a complete list of authors.
// Licensed under the GNU LESSER GENERAL PUBLIC LICENSE. See license.txt for details.


// Initialization
require_once('tiki-setup.php');
include_once('lib/dcs/dcslib.php');

if(!isset($dcslib)) {
    $dcslib= new DCSLib($dbTiki);
}


function db_to_form($date){
    $d = array();

      $d[day] = substr($date, 8, 2);
      $d[month] = substr($date, 5, 2);
      $d[year] = substr($date, 0, 4);

      return $d[day]."/".$d[month]."/".$d[year];
}


function unite($prod_num) {
  extract($GLOBALS);
  $query = "select * from `PRODUIT` where PROD_NUM='".$prod_num."'" ;
  $resultat1 = $tikilib->query($query);
  $resultat2=$resultat1->fetchRow(DB_FETCHMODE_ASSOC);
  return $resultat2["UNITE"];
}

function nom_produit($prod_num) {
  extract($GLOBALS);
  $query = "select * from `PRODUIT` where PROD_NUM='".$prod_num."'" ;
  $resultat1 = $tikilib->query($query);
  $resultat2=$resultat1->fetchRow(DB_FETCHMODE_ASSOC);
  return $resultat2["NOM"];
}

function nom_usine($fab_num) {
  extract($GLOBALS);
  $query = "select * from `USINE` where FAB_NUM='".$fab_num."'" ;
  $resultat1 = $tikilib->query($query);
  $resultat2=$resultat1->fetchRow(DB_FETCHMODE_ASSOC);
  return $resultat2["NOM"];
}

function nom_depot($dep_num) {
  extract($GLOBALS);
  $query = "select * from `DEPOT` where DEP_NUM='".$dep_num."'" ;
  $resultat1 = $tikilib->query($query);
  $resultat2=$resultat1->fetchRow(DB_FETCHMODE_ASSOC);
  return $resultat2["NOM"];
}

function matieres_premieres ($prod_num) {
  extract ($GLOBALS);
```

```
  $resultat2 ="";
  $query = "select * from ‘TRANSFORMATION‘ where SORTANT=’".$prod_num."’" ;
   $resultat1 = $tikilib->query($query);
  while($ligne = $resultat1->fetchRow(DB_FETCHMODE_ASSOC)) {
    $resultat2= $resultat2.$ligne["QUANTITE_INTRANT"].
               " ".unite($ligne["INTRANT"])." ".nom_produit($ligne["INTRANT"])."<br>";
  }
  return $resultat2;
  }

function produits_commande ($com_num) {
  extract ($GLOBALS);
  $resultat2 ="";
  $query = "select * from ‘CONTENU_COMMANDE‘ where COM_NUM=’".$com_num."’" ;
   $resultat1 = $tikilib->query($query);
  while($ligne = $resultat1->fetchRow(DB_FETCHMODE_ASSOC)) {
    $resultat2= $resultat2.$ligne["QUANTITE"].
                " ".unite($ligne["PROD_NUM"])." ".nom_produit($ligne["PROD_NUM"])."<br>";
  }
  return $resultat2;
  }

function prix($prod_num) {
  extract ($GLOBALS);
  $query = "select * from ‘PRODUIT‘ where PROD_NUM=’".$prod_num."’" ;
  $resultat1 = $tikilib->query($query);
  $resultat2=$resultat1->fetchRow(DB_FETCHMODE_ASSOC);
  return $resultat2["PRIX"];
}

function is_between_dates($com_num, $date1, $date2) {
  extract ($GLOBALS);
  $resultat2 ="";
  $query = "select * from ‘COMMANDE‘ where
           COM_NUM=’".$com_num."’ and COM_DATE BETWEEN ’".$date1."’ AND ’".$date2."’" ;
  $resultat1 = $tikilib->query($query);
  $resultat2=$resultat1->fetchRow(DB_FETCHMODE_ASSOC);
  if ($resultat2["COM_NUM"]==$com_num) {return ’y’;}
  return ’n’;
}


function nom_usine_commande ($com_num) {
  extract ($GLOBALS);
  $resultat2 ="";
  $query = "select * from ‘COMMANDE‘ where COM_NUM=’".$com_num."’" ;
  $resultat1 = $tikilib->query($query);
  $resultat2=$resultat1->fetchRow(DB_FETCHMODE_ASSOC);
  return nom_usine($resultat2["FAB_NUM"]);
  }

if (isset ( $_REQUEST["afficher_produits"])){

  $query = "select * from ‘PRODUIT‘";
   $tableau_produit = "<table cellspacing = 1pt cellpadding=10pt border=1pt>
                       <tr><td><b>NUMERO</b></td>
                       <td><b>NOM</b></<td>
                       <td><b>UNITE DE MESURE</b></td>
                       <td><b>PRIX</b></td>
                       <td><b>INTRANTS</b></tr>";
  $listing_complet = $tikilib->query($query);
  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_produit=$tableau_produit."<tr>
                     <td>".$ligne["PROD_NUM"]."</td>
                     <td>".$ligne["NOM"]."</td>
```

```
                    <td>".$ligne["UNITE"]."</td>
                    <td>".$ligne["PRIX"]." eur</td>
                    <TD>".matieres_premieres($ligne["PROD_NUM"])."</td></tr>";
  }
  $tableau_produit=$tableau_produit."</table>";
$smarty->assign ('tableau', $tableau_produit);
 $smarty->assign ('ajout', "produit");
}

if (isset ( $_REQUEST["afficher_usines"])){

  $query = "select * from ‘USINE‘";
   $tableau_usine = "<table cellspacing = 1pt cellpadding=10pt border=1pt>
                   <tr><td><b>NUMERO</b></td>
                   <td><b>NOM</b></<td>
                   <td><b>LOCALISATION</b></td>
                   <td><b>CAPACITE TOTALE DE PRODUCTION</b></td></tr>";
  $listing_complet = $tikilib->query($query);

  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_usine=$tableau_usine."<tr>
                  <td>".$ligne["FAB_NUM"]."</td>
                  <td>".$ligne["NOM"]."</td>
                  <td>".$ligne["LOCALISATION"]."</td>
                  <td>".$ligne["CAPA_PROD"]."</td></tr>";
  }
  $tableau_usine=$tableau_usine."</table>";
$smarty->assign ('tableau', $tableau_usine);
 $smarty->assign ('ajout', "(e)_usine");


}

if (isset ( $_REQUEST["afficher_depots"])){

  $query = "select * from ‘DEPOT‘";
   $tableau_depot = "<table cellspacing = 1pt cellpadding=10pt border=1pt>
                   <tr><td><b>NUMERO</b></td>
                   <td><b>NOM</b></<td>
                   <td><b>LOCALISATION</b></td>
                   <td><b>CAPACITE TOTALE DE STOCKAGE</b></td></tr>";
  $listing_complet = $tikilib->query($query);
  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_depot=$tableau_depot."<tr>
                  <td>".$ligne["DEP_NUM"]."</td>
                  <td>".$ligne["NOM"]."</td>
                  <td>".$ligne["LOCALISATION"]."</td>
                  <td>".$ligne["CAPA_STOCK"]."</td></tr>";
  }
  $tableau_depot=$tableau_depot."</table>";
$smarty->assign ('tableau', $tableau_depot);
 $smarty->assign ('ajout', "depot");


}

if (isset ( $_REQUEST["afficher_commandes"])){

  $query = "select * from ‘COMMANDE‘";
   $tableau_commande = "<table cellspacing = 1pt cellpadding=10pt border=1pt><tr>
                       <td><b>NUMERO</b></td>
                       <td><b>DATE</b></<td>
                       <td><b>FABRIQUE QUI PASSE COMMANDE</b></td>
                       <td><b>DEPOT QUI FOURNIT COMMANDE</b></td>
                       <td><b>PRODUITS COMMANDES</b></td></tr>";
```

```
  $listing_complet = $tikilib->query($query);
  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_commande=$tableau_commande."<tr>
                      <td>".$ligne["COM_NUM"]."</td>
                      <td>".db_to_form($ligne["COM_DATE"])."</td>
                      <td>".nom_usine($ligne["FAB_NUM"])."</td>
                      <td>".nom_depot($ligne["DEP_NUM"])."</td>
                      <td>".produits_commande($ligne["COM_NUM"])."</td></tr>";
  }
  $tableau_commande=$tableau_commande."</table>";
$smarty->assign ('tableau', $tableau_commande);
 $smarty->assign ('ajout', "(e)_commande");
}


if (isset ( $_REQUEST["afficher_produits_fabriques"])){

  $query = "select * from ‘PRODUIT_FABRIQUE‘";
   $tableau_produit_fabrique = "<table cellspacing = 1pt cellpadding=10pt border=1pt><tr>
                               <td><b>NUMERO FABRIQUE</b></td>
                               <td><b>NUMERO PRODUIT</b></<td>
                               <td><b>CAPACITE MINIMALE DE PRODUCTION</b></td>
                               <td><b>CAPACITE MAXIMALE DE PRODUCTION</b></td>
                               <TD><B>PRODUCTION ACTUELLE</B></TD></tr>";
  $listing_complet = $tikilib->query($query);
  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_produit_fabrique=$tableau_produit_fabrique."<tr>
                              <td>".nom_usine($ligne["FAB_NUM"])."</td>
                              <td>".nom_produit($ligne["PROD_NUM"])."</td>
                              <td>".$ligne["MIN_FAB"]."</td>
                              <td>".$ligne["MAX_FAB"]."</td>.
                              <TD>".$ligne["FAB_ACTUELLE"]." ".
                                   unite($ligne["PROD_NUM"])."</TD></tr>";
  }
  $tableau_produit_fabrique=$tableau_produit_fabrique."</table>";
$smarty->assign ('tableau', $tableau_produit_fabrique);
 $smarty->assign ('ajout', "produit_en_fabrication");

}

if (isset ( $_REQUEST["afficher_produits_stockes"])){

  $query = "select * from ‘PRODUIT_STOCKE‘";
   $tableau_produit_stocke = "<table cellspacing = 1pt cellpadding=10pt border=1pt><tr>
                             <td><b>NUMERO DEPOT</b></td>
                             <td><b>NUMERO PRODUIT</b></<td>
                             <td><b>QUANTITE</b></td></tr>";
  $listing_complet = $tikilib->query($query);
  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_produit_stocke=$tableau_produit_stocke."<tr>
                            <td>".nom_depot($ligne["DEP_NUM"])."</td>
                            <td>".nom_produit($ligne["PROD_NUM"])."</td>
                            <td>".$ligne["QUANTITE_DUE"]." ".
                                  unite($ligne["PROD_NUM"])."</td></tr>";
  }
  $tableau_produit_stocke=$tableau_produit_stocke."</table>";
$smarty->assign ('tableau', $tableau_produit_stocke);
 $smarty->assign ('ajout', "produit_en_stock");
}


if (isset ($_REQUEST["ajouter_produit"])) {
 $num = $_REQUEST["num"];
 $nom = $_REQUEST["nom"];
 $unite = $_REQUEST["unite"];
```

```
 $prix = $_REQUEST["prix"];
  $query1 = "INSERT INTO ‘PRODUIT‘ VALUES (’$num’, ’$nom’, ’$unite’, ’$prix’)";

  $tikilib->query($query1);
 $query = "select * from ‘PRODUIT‘";
   $tableau_produit = "<table cellspacing = 1pt cellpadding=10pt border=1pt><tr>
                       <td><b>NUMERO</b></td>
                       <td><b>NOM</b></<td>
                       <td><b>UNITE DE MESURE</b></td>
                       <td><b>PRIX</b></td>
                       <td><b>INTRANTS</b></tr>";
  $listing_complet = $tikilib->query($query);
  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_produit=$tableau_produit."<tr>
                      <td>".$ligne["PROD_NUM"]."</td>
                      <td>".$ligne["NOM"]."</td>
                      <td>".$ligne["UNITE"]."</td>
                      <td>".$ligne["PRIX"]." eur</td>
                      <TD>".matieres_premieres($ligne["PROD_NUM"])."</td></tr>";
  }
  $tableau_produit=$tableau_produit."</table>";
$smarty->assign (’tableau’, $tableau_produit);
 $smarty->assign (’ajout’, "produit");

}

if (isset ($_REQUEST["requete_speciale1"])) {
  $query = "select * from ‘PRODUIT‘";
   $tableau_produit = "<h4>Cliquez sur un produit pour l’analyser:</h4><br>
                       "."<table cellspacing = 1pt cellpadding=10pt border=1pt><tr>
                       <td><b>NUMERO</b></td>
                       <td><b>NOM</b></<td>
                       <td><b>UNITE DE MESURE</b></td>
                       <td><b>PRIX</b></td>
                       <td><b>INTRANTS</b></tr>";
  $listing_complet = $tikilib->query($query);
  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_produit=$tableau_produit."<tr>
                      <td>"."<a href=\"tiki-db2.php?requete_speciale1suite&num=".
                           $ligne["PROD_NUM"]."\">".$ligne["PROD_NUM"]."</a></td>
                      <td>".$ligne["NOM"]."</td>
                      <td>".$ligne["UNITE"]."</td>
                      <td>".$ligne["PRIX"]." eur</td>
                      <TD>".matieres_premieres($ligne["PROD_NUM"])."</td></tr>";
  }
  $tableau_produit=$tableau_produit."</table>";
$smarty->assign (’tableau’, $tableau_produit);
 $smarty->assign (’ajout’, "");
}

if (isset ($_REQUEST["requete_speciale1suite"])) {
  $num = $_REQUEST["num"];
  $query = "select * from ‘CONTENU_COMMANDE‘where PROD_NUM=’".$num."’";
  $listing = $tikilib->query ($query);
  $tableau_res = "<h4>Identifiant des lignes de commande et nom des unites
                       de fabrication pour le produit: ".nom_produit($num)."</h4>
                 <table cellspacing =1pt cellpadding=10pt border=1pt><tr>
                 <td><b>Numero ligne de commande</b></td>
                 <td><b>Nom de l’unite de fabrication</b></td></tr>";
  while ($ligne = $listing->fetchRow (DB_FETCHMODE_ASSOC)) {
    $tableau_res = $tableau_res."<tr>
                    <td>".$ligne["COM_NUM"]. "</td>
                    <td>".nom_usine_commande($ligne["COM_NUM"])."</td></tr>";
  }
  $tableau_res=$tableau_res."</table>";
```

```
  $smarty->assign ('tableau', $tableau_res );
  $smarty->assign ('ajout', "");
}

if (isset($_REQUEST["requete_speciale2"])) {
$formulaire="<h4>Choisissez vos parametres</h4>
             <form action=\"tiki-db2.php\" method=\"post\">
             Numero produit: <input type=\"text\" name=\"num\" size=2> <br>
             Date depart (format YYYY-MM-DD): <input type=\"text\"
                                 name=\"date1\" size=10> <br>
             Date fin format (YYYY-MM-DD): <input type=\"text\"
                                 name=\"date2\" size=10> <br>
             <input type=\"submit\" name=\"calculer_montant\"
                         value=\"Calculer le montant total\"> </form>";
  $smarty->assign ('tableau', $formulaire);
  $smarty->assign ('ajout', '');
}

if (isset($_REQUEST["calculer_montant"])) {
  $num = $_REQUEST["num"];
  $date1 = $_REQUEST["date1"];
  $date2 = $_REQUEST["date2"];
  $resultat2=0;
  $query = "select * from `CONTENU_COMMANDE` where PROD_NUM='".$num."'" ;
   $resultat1 = $tikilib->query($query);
  while($ligne = $resultat1->fetchRow(DB_FETCHMODE_ASSOC)) {

     if (is_between_dates ($ligne["COM_NUM"],$date1,$date2)=='y'){
            $resultat2=$resultat2 + ((0+$ligne["QUANTITE"])*(0+prix($ligne["PROD_NUM"])));
       }
  }
  $resultat3="Le resultat total est: ".strval($resultat2)." EUR ";
  $smarty->assign ('tableau', $resultat3);
  $smarty->assign ('ajout', "");
}


if (isset($_REQUEST["requete_speciale3"])) {
  $query = "select * from `PRODUIT`";
   $tableau_produit = "<table cellspacing = 1pt cellpadding=10pt border=1pt><tr>
                     <td><b>NUMERO</b></td>
                     <td><b>NOM</b></<td>
                     <td><b>UNITE DE MESURE</b></td>
                     <td><b>PRIX</b></td>
                     <td><b>INTRANTS</b></tr>";
  $listing_complet = $tikilib->query($query);
  while($ligne = $listing_complet->fetchRow(DB_FETCHMODE_ASSOC)) {
    $tableau_produit=$tableau_produit."<tr>
                    <td>".$ligne["PROD_NUM"]."</td>
                    <td>".$ligne["NOM"]."</td>
                    <td>".$ligne["UNITE"]."</td>
                    <td>".$ligne["PRIX"]." eur</td>
                    <TD>".matieres_premieres($ligne["PROD_NUM"])."</td></tr>";
  }
  $tableau_produit=$tableau_produit."</table>";
$smarty->assign ('tableau', $tableau_produit);
 $smarty->assign ('ajout', "");
}


$smarty->assign( 'mid', 'tiki-db2.tpl' );
$smarty->display( "styles/$style_base/tiki.tpl" );

?>
```